KATHOLIEKE UNIVERSITEIT
LEUVEN

Faculty of Economics and Applied Economics

Timeslack-based techniques for generating robust project schedules subject to resource uncertainty

Olivier Lambrechts, Erik Demeulemeester and Willy Herroelen

DEPARTMENT OF DECISION SCIENCES AND INFORMATION MANAGEMENT (KBI)

KBI 0713

Time slack-based techniques for generating robust project schedules subject to resource uncertainty*

Olivier Lambrechts Erik Demeulemeester

Willy Herroelen†

Research Center for Operations Management

Department of Decision Sciences and Information Management

Faculty of Economics and Applied Economics

Katholieke Universiteit Leuven (Belgium)

Abstract

The classical, deterministic resource-constrained project scheduling problem has been the subject of a great deal of research during the previous decades. This is not surprising given the high practical relevance of this scheduling problem. Nevertheless, extensions are needed to be better able to cope with situations arising in practice such as multiple activity execution modes, activity duration changes and resource breakdowns. In this paper we analytically determine the

*This research has been supported by project OT/03/14 of the Research Fund of K.U.Leuven, project G.0109.04 of the Research Programme of the Fund for Scientific Research - Flanders (Belgium) (F.W.O.-Vlaanderen) and project NB/06/06 of the National Bank of Belgium.

†Corresponding author. Tel: +32-16-326970; fax: +32-16-326732; e-mail: *willy.herroelen@econ.kuleuven.be*

impact of unexpected resource breakdowns on activity durations. Furthermore, using this information we develop an approach for inserting explicit idle time into the project schedule in order to protect it as well as possible from disruptions caused by resource unavailabilities. This strategy will be compared to a traditional simulation-based procedure and to a heuristic developed for the case of stochastic activity durations.

1 Introduction

Most of the research in project scheduling deals with the generation of an *initial project schedule* (*baseline schedule*) in a static and deterministic environment with complete information. For an extensive overview we refer to Brucker et al. (1999), Herroelen et al. (1998) and Demeulemeester and Herroelen (2002).

Unfortunately, these underlying assumptions simply do not always hold in practice. In the real world, a project manager often has to deal with a stochastic and dynamic scheduling environment. He has to protect the initial baseline schedule from the adverse effects of possible disruptions because often project activities are subcontracted or executed by resources that are not exclusively reserved for the current project. A change in the starting times of such activities could lead to additional costs due to required subcontractor flexibility and due to schedule nervousness. A possible measure for the deviation between the initial schedule and the realized schedule is the *weighted instability cost*. It can be calculated by taking the sum of the expected weighted absolute deviations between the planned and the actually realized activity starting times. The weight w_i, assigned to each activity i, reflects that activity's importance of starting it at its planned starting time in the initial schedule. More specifically, w_i denotes the marginal cost of deviating from the planned starting time of activity i during project execution. Recent research by Leus (2004), by Herroelen and Leus (2004), by

Leus and Herroelen (2004) and by Van de Vonder et al. (2005, 2006, 2007b, 2007c) considers this objective function for the case of project scheduling with stochastic activity durations. Other possible causes for uncertainty in project execution might be, amongst others, inaccurate time estimates, bad weather conditions or unavailability of resources. In this paper we study the last of these possible causes.

Resource breakdowns have been cited by numerous authors as one of the most important sources of disruptions in practical project management (see amongst others Yu and Qi (2004)). We only consider renewable resources. This means that each resource type k ($k : 1 \rightarrow R$) is modeled as a set of individual resource units. In the deterministic case these resource units are assumed to be available throughout the project on a period-per-period basis. In the stochastic case, on the other hand, breakdowns may occur. Whenever a resource unit breaks down, it has to be repaired before it becomes available again. The time between the end of a repair period and a new failure for resource unit m of resource type k is modeled by means of a stochastic variable $\boldsymbol{X}_{mk}$. The time needed to repair a resource unit m of type k is also represented by means of a stochastic variable $\boldsymbol{Y}_{mk}$.

In order to cope with uncertainty, one has several options at one's disposal. In their excellent overview paper on scheduling under uncertainty, Davenport and Beck (2002) distinguish between proactive and reactive scheduling. *Proactive scheduling* focuses on the construction of predictive schedules that use statistical knowledge of the uncertainties with the aim of increasing schedule robustness. A schedule is considered to be robust if it can absorb anticipated disruptions without affecting planned external activities while maintaining high shop performance (O'Donovan et al., 1999). Approaches to build such a robust schedule can be based on redundancy, probabilistic techniques or contingent scheduling. In this paper we focus on the construction of robust project schedules based on redundancy. This implies the reservation of extra time and/or resource capacity so that unexpected events

during execution can be absorbed by these time and/or resource buffers. Unfortunately, no matter how much care is taken in constructing a proactive schedule, disruptions can never be totally prevented.

In case an activity is delayed due to for example an unforeseen resource breakdown, the schedule may become infeasible. A reactive procedure must then be used to repair the schedule. The aim of this reactive procedure is to restore schedule feasibility in such a way that some objective function (such as the deviation from the baseline schedule) is optimized.

Here, we only focus on the construction of a robust predictive schedule. For more information regarding the reactive phase we would like to refer the interested reader to Van de Vonder et al. (2007a) for the stochastic duration case and to Lambrechts et al. (2007a) for the stochastic resource availability case.

The next section will briefly introduce our scheduling problem together with a number of definitions and concepts that will be of importance in the remainder of the paper. In Section 3 we will analytically show how resource breakdowns can be translated into activity duration increases under various assumptions. This information will be used in Section 4 to develop an approach for strategically inserting explicit idle time into the schedule in order to minimize the total instability cost. In a computational experiment this approach is compared with a simulation-based approach and with a dedicated approach for minimizing instability in case of stochastic activity durations. The results of this experiment are given in Section 5. Finally, we terminate with some concluding remarks and suggestions for further research.

2 Problem statement

The aim of the proactive baseline scheduling problem is to generate a project schedule that is feasible as well as robust. If we assume that the project is represented using the activity-on-node representation, the digraph $G =$

(N, A) contains a set of nodes N and a set of arcs A. The nodes represent the activities constituting the project whereas the arcs represent the finish-start, zero-lag precedence relations. Whenever $(i, j) \in A$ we say that activity i (i : 1 to n) is an immediate predecessor of activity j, implying that activity j cannot start before activity i has finished:

$$s_i + d_i \leqslant s_j \qquad \forall (i, j) \in A \tag{2.1}$$

with s_i representing the starting time of activity i and d_i the deterministic duration of activity i.

In resource-constrained project scheduling we also have to take the renewable resource constraints into account. As we indicated in Section 1, we assume that a finite amount a_k of each resource type k is available on a period-per-period basis. Resource feasibility then implies that for each time period t and for each resource type k the sum of the resource requirements r_{ik} of the activities that are in progress during period t ($\mathcal{S}_t$) cannot exceed the availability a_k:

$$\sum_{i \in \mathcal{S}_t} r_{ik} \leqslant a_k \qquad \forall t, \forall k \tag{2.2}$$

Finally, a given project deadline δ has to be respected:

$$s_n \leqslant \delta \tag{2.3}$$

with n the dummy end activity having a duration and a resource usage equal to 0 and representing project completion.

Our objective then becomes the generation of a baseline schedule depicted by means of a vector of starting times $S^0 = (s_1^0, ..., s_n^0)$ respecting the constraints 2.1, 2.2 and 2.3 and minimizing the following objective function:

$$\sum_{i \in N} w_i |E(\mathbf{s}_i) - s_i^0| \tag{2.4}$$

Because of the stochastic nature of the problem we cannot always stick to this baseline schedule. The real starting times are consequently stochastic variables that are represented by the stochastic vector $\mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n)$ and that depend on the realization of the stochastic resource availabilities ($\mathbf{a}_{kt}$), on the planned starting times (s_i) and on the reactive policy $\mathcal{R}$ that is used to repair a disrupted schedule. We assume that a 'railroad scheduling' approach is used, meaning that activities are never started before their planned starting time ($\mathbf{s}_i \geqslant s_i$). Not imposing this constraint would render the inclusion of explicit idle time pointless for minimizing schedule instability.

In this paper, we solely focus on the construction of the baseline schedule S^0. We assume that an initial, unbuffered baseline schedule S^u is given. Our aim will be to improve the robustness of this schedule by inserting explicit idle time into the project schedule. In order to be able to do this in an efficient way, we need a more thorough understanding of the nature of resource breakdowns and their impact on the duration of the disrupted activity. This impact will be studied in the next section. First we need to introduce some key concepts regarding resource breakdowns and activity interruptions.

2.1 Preempt-repeat versus preempt-resume

In case an activity is interrupted due to for example a resource breakdown, it will either have to be restarted from scratch or it will simply be resumed from the point where execution was halted. The first case is called *preempt-repeat*, the second case is called *preempt-resume*.

Preempt-repeat implies that all the time and effort that was invested in the execution of that activity until the time of the interruption is lost. This scenario is encountered in practice whenever an activity must be executed without interruption. An example is mixing concrete. If the mixer fails before the concrete is used, the mixture will dry, rendering it useless for further building activities and necessitating the creation of a new load of concrete.

In practice, more often than not, the preempt-resume setting will be more applicable. This implies that whenever an activity is interrupted and preempted, it can be continued from the point where execution was halted whenever the reason for the interruption (in our case the machine breakdown) is removed. An example could be the excavation of a building site. Dragline excavators are often used for such activities. In case an excavator fails, the excavation can simply be continued from the point where it was interrupted after repair or replacement of the excavator.

Of course, both cases are often a simplification of reality. It can be imagined that in practice a mixed form is more likely. Usually, activities will not have to be restarted all the way from zero after they were preempted but it will probably also not be possible to carry on as if nothing happened. The third possibility is therefore that whenever an activity is preempted, a setup time has to be taken into account when restarting this activity. Therefore we call this variant *preempt-setup*. This setup time is defined relative to the activity's duration. It does not seem unreasonable to assume that activities that take longer to complete are likely to be more complicated and will therefore require more setup time. We can then write:

$$\text{setup time}_i = \varsigma_i d_i \tag{2.5}$$

We call ς_i the setup factor of activity i.

2.2 Modeling times to failure and repair times

A lot has been written about the choice of an appropriate distribution function for modeling the time to failure of a resource (Barlow and Proschan, 1996).

The use of the exponential distribution is supported by empirical evidence as well as by mathematical arguments. This can be motivated as follows: complex resources can fail for a wide variety of reasons. We can

therefore consider each resource unit to be composed of different components, each associated with a possible failure cause, with different times to failure. Let $\mathbf{N}(t)$ be the total amount of breakdowns up to time t, split up by components 1 to m as follows: $\mathbf{N}(t) = \mathbf{N}_1(t) + \mathbf{N}_2(t) + ... + \mathbf{N}_m(t)$. If m is large enough and the times between counts for each breakdown cause are independent and identically distributed stochastic variables, then the resulting counting process $\mathbf{N}(t)$ will follow a Poisson distribution (Hopp and Spearman, 2001). Because a Poisson counting process corresponds to an exponential distribution of interarrival times (Girault, 1959), the times between failures will be exponentially distributed. It is true that other failure distributions may be more suitable for certain equipment types, but for most project planning cases it seems safe to assume that the resources that really matter in the scheduling process are expensive and therefore often complex. A clear advantage of using the exponential distribution is that it is unambiguously defined by its expected value. This means that we only need to know the mean time to failure for each resource type k ($MTTF_k$) to know the failure distribution function. A similar choice was made by O'Donovan et al. (1999) for modeling interfailure times when processing jobs on a single machine subject to breakdowns.

Unfortunately, this reasoning cannot be so easily applied to the repair times. However, it is analytically interesting but also practically acceptable to assume that these times are also exponentially distributed. This assumption is not entirely unrealistic. Complex machinery can fail for a wide variety of reasons and logically, not all these causes are equally easy or hard to remedy. Overall it can be expected that the probability that a major problem, with a long repair time, occurs is low whereas the occurrence of simple to solve problems is more frequent. An exponential distribution would therefore seem suitable. Even the fact that the exponential distribution is not bounded is not necessarily problematic. In rare occasions, a disruption of a critical resource could possibly compromise the feasibility of the project as a

whole. Imagine for example the explosion of a shuttle in the course of a space exploration project and the potentially resulting shutdown of fund flow for this project. Nevertheless, we also consider the uniform distribution as an alternative repair time distribution function which may be more appropriate in certain cases.

2.3 Free versus fixed resource allocations

When constructing a project baseline schedule, the decision maker has to decide on the starting times for each activity and consequently also on the amount of resource units of each resource type required in each time period. The project manager can also decide to allocate specific resource units to individual activities in advance. The result of this decision will be that rescheduling becomes far easier as the affected activities simply have to be right-shifted to restore schedule feasibility. Furthermore, it will allow us to analytically determine the expected duration increase for each activity due to resource breakdowns as shown in Section 3.

A small example is now given to illustrate the impact of resource breakdowns and fixed allocations on project execution. Consider the situation depicted in the top schedule of Figure 1. Activity i has a deterministic duration $d_i = 6$ and a resource usage $r_i = 3$ of a single renewable resource type with a per period availability of 6. Each of the r_i resource units that are used for executing activity i are subject to resource breakdowns. In order to estimate the impact on project stability of potential breakdowns, we are interested in the expected increase of the duration of activity i due to these breakdowns. Imagine the situation depicted in the bottom schedule of Figure 1. We see that resource unit 2 experiences a breakdown after 6 time periods ($X_{12} = 6$) and that its repair takes 2 time periods ($Y_{12} = 2$). Resource allocations are fixed meaning that only the activity that is using resource unit 2 between time points 6 and 8 is affected. The result will be a preemption of activity i until time point 8. In case of preempt-repeat, this means that

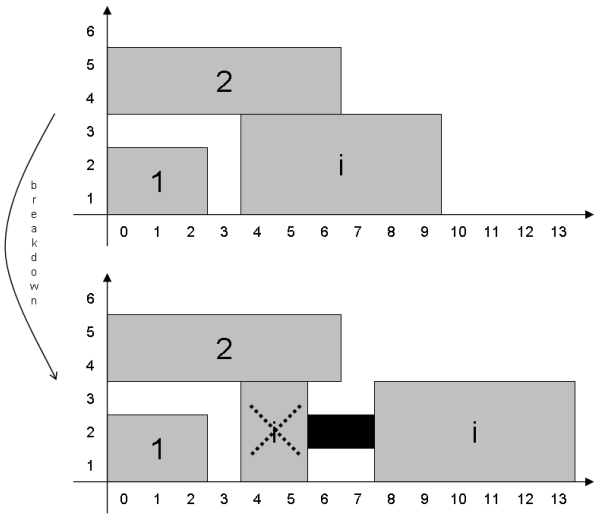

Figure 1: Impact of a resource breakdown

activity i will have to be restarted from scratch after the resource unit is repaired.

Summarized, this means that in case activity i requires r_i resource units, only these specific resource units will be used for executing i and a breakdown of one or more of these units translates directly into an interruption of i.

The main drawback of this assumption is that practically all rescheduling flexibility will be lost because idle resource units nor resource units allocated to less important activities can be used to overcome a possible resource shortage for activity i. In our example, the duration increase of activity i due to the breakdown of resource unit 2 could easily have been prevented by replacing it with resource unit 6 in case resource allocations were free. However, omitting this assumption renders the calculation of expected duration increases practically impossible.

Resource allocations can be fixed using the approach of resource flow networks. A resource flow network implies an extension of the set of arcs A with the resource flow arcs A_R. The resource flow arcs are precedence relations that are added in order to fix resource allocations and that eliminate

explicit resource constraints.

Artigues and Roubellat (2000) introduce a simple method for generating a resource flow network by extending a parallel schedule generation scheme to derive the flows. The main advantage of this approach is that it is very fast but unfortunately it does not take schedule robustness into account. Luckily this drawback does not have a significant impact on our final schedule since time buffering is able to compensate for the deficiencies of a non-robust resource allocation as was shown by Van de Vonder (2006). Furthermore, we only fix resource allocations during the time buffering step but release them during schedule execution in order not to unnecessarily restrict rescheduling flexibility.

3 Breakdown process

In this section, we will show how resource breakdowns affect an activity's real duration under various scenarios. Throughout this analysis, we will assume that the resources are allocated to the activities constituting the project before project execution starts and that this resource allocation remains fixed until the project finishes. Furthermore, we assume that the time to failure and the time to repair of each resource unit m of resource type k follows the same distribution function, that the interfailure times and the repair times are mutually independent and that resource units can only break down when they are in use.

The *preempt-repeat* case is studied in section 3.1, the *preempt-resume* case in section 3.2. In each section, we first analyze the impact of resource breakdowns on an activity's duration in case only one resource type is used to execute that activity. This analysis will then be extended to deal with multiple resource types. Note that we do not make any assumptions regarding failure time or repair time distributions until the end of each subsection in which results are derived for specific distribution functions.

3.1 Preempt-Repeat

3.1.1 Single resource type

Because of interruptions such as resource breakdowns, the real duration of activity i becomes a stochastic variable $\boldsymbol{d}'_i$ consisting of a deterministic part d_i, corresponding to the duration of the activity when no interruption occurs and after which i is terminated, and a stochastic part $\boldsymbol{\sigma}_i$, corresponding to the total failed execution time (i.e. not resulting in activity completion) $\boldsymbol{\mathcal{X}}_i$ together with the total repair time $\boldsymbol{\mathcal{Y}}_i$. If we denote the length of the r'th failed execution or repair time as $\boldsymbol{\mathfrak{F}}_{ir}$, respectively $\boldsymbol{\mathfrak{R}}_{ir}$, and the number of interruptions as $\boldsymbol{\mathcal{N}}_i$, we can define:

$$\begin{aligned} \boldsymbol{d}'_i &= d_i + \boldsymbol{\sigma}_i && (3.1)\\ \boldsymbol{\sigma}_i &= \boldsymbol{\mathcal{X}}_i + \boldsymbol{\mathcal{Y}}_i && (3.2)\\ \boldsymbol{\mathcal{X}}_i &= \boldsymbol{\mathfrak{F}}_{i1} + ... + \boldsymbol{\mathfrak{F}}_{i\boldsymbol{\mathcal{N}}_i} && (3.3)\\ \boldsymbol{\mathcal{Y}}_i &= \boldsymbol{\mathfrak{R}}_{i1} + ... + \boldsymbol{\mathfrak{R}}_{i\boldsymbol{\mathcal{N}}_i} && (3.4) \end{aligned}$$

We can calculate the expected value of $\boldsymbol{\sigma}_i$ as follows:

$$\begin{aligned} E[\boldsymbol{\sigma}_i] &= E[\boldsymbol{\mathcal{X}}_i] + E[\boldsymbol{\mathcal{Y}}_i] \\ &= E[\boldsymbol{\mathcal{N}}_i]E[\boldsymbol{\mathfrak{F}}_i] + E[\boldsymbol{\mathcal{N}}_i]E[\boldsymbol{\mathfrak{R}}_i] && (3.5) \end{aligned}$$

because we assume that the interfailure times are independent from the repair times. Note that this does unfortunately not hold for the interfailure times and the number of interruptions. Nevertheless, simulation results show that assuming independence in our equation does not significantly alter the results. Furthermore, we assume a constant failure rate (i.e. breakdowns do not become more or less likely right after resources are restarted for activity re-execution).

In case we only consider one resource type then the time to restart execution of an interrupted activity i is equal to the time to repair a resource unit used by that activity i or $\mathfrak{R}_i = \boldsymbol{Y}$ (assuming that all resource units of that single resource type have the same distribution function determining the time to repair $\boldsymbol{Y}$). Note that this does not hold for $\mathfrak{F}_i$. $\mathfrak{F}_i$ differs from $\boldsymbol{X}$ in two important respects. First of all, whereas $\boldsymbol{X}$ represents the time to failure of a single resource unit, $\mathfrak{F}_i$ represents the minimum time to failure over all of the r_i resource units allocated to i. This distinction is important because i is supposed to be interrupted as soon as one of the resource units used to execute i breaks down. Secondly, whereas $\boldsymbol{X}$ is able to take on values larger than d_i, this would clearly not make any sense in our analysis as this does not correspond to a failed execution but to a completion of i and the probability density function (*pdf*) should therefore be conditioned on this fact. The distribution function of the minimum of a number of independently and identically distributed random variables can be determined using lemma 1. This distribution function can then be modified into a conditional distribution function constrained between 0 and d_i as follows (Blumenfeld, 2001):

$$f'(x'|x' < d_i) \quad = \quad \frac{f(x')}{Pr(x' < d_i)} \tag{3.6}$$

Lemma 1. *Let $\boldsymbol{X}_1$, $\boldsymbol{X}_2$, ..., $\boldsymbol{X}_n$ be independent and identically distributed stochastic variables with cdf (cumulative distribution function) $F(x)$. The minimum of these variables, $\boldsymbol{Z}$ with cdf $G(z)$, will then be distributed as follows:*

$$G(z) = 1 - [1 - F(z)]^n \tag{3.7}$$

Let us first analyze the stochastic variable describing the number of interruptions experienced by activity i throughout its execution ($\boldsymbol{\mathcal{N}}_i$). The expected value of $\boldsymbol{\mathcal{N}}_i$ can be calculated using lemma 2.

Lemma 2. *If we let ψ_i represent the probability that activity i is interrupted in a preempt-repeat scenario, the expected number of interruptions until i finishes is given by:*

$$E[\boldsymbol{\mathcal{N}}_i] = \frac{\psi_i}{1-\psi_i} \tag{3.8}$$

Proof. If ψ_i represents the probability that activity i is interrupted then the number of interruptions is obviously distributed with *pdf*:

$$h(\mathcal{N}_i) \;=\; (1-\psi_i)\psi_i^{\mathcal{N}_i} \qquad (\mathcal{N}_i = 0, 1, 2, ...) \tag{3.9}$$

and expected value:

$$\begin{aligned} E(\boldsymbol{\mathcal{N}_i}) &= \sum_{\mathcal{N}_i=0}^{\infty} \mathcal{N}_i(1-\psi_i)\psi_i^{\mathcal{N}_i} \\ &= (1-\psi_i)\sum_{\mathcal{N}_i=0}^{\infty} \mathcal{N}_i\psi_i^{\mathcal{N}_i} \end{aligned}$$

A closed form expression for the infinite sum in the last equation can be determined as follows:

$$\begin{aligned} \sum_{\mathcal{N}_i=0}^{\infty} \mathcal{N}_i\psi_i^{\mathcal{N}_i} &= \psi_i + 2\psi_i^2 + 3\psi_i^3 + 4\psi_i^4 + ... \\ \psi_i \sum_{\mathcal{N}_i=0}^{\infty} \mathcal{N}_i\psi_i^{\mathcal{N}_i} &= \psi_i^2 + 2\psi_i^3 + 3\psi_i^4 + 4\psi_i^5 + ... \\ (1-\psi_i) \sum_{\mathcal{N}_i=0}^{\infty} \mathcal{N}_i\psi_i^{\mathcal{N}_i} &= \psi_i + \psi_i^2 + \psi_i^3 + \psi_i^4 + \psi_i^5 + ... \end{aligned}$$

Furthmore

$$\sum_{\mathcal{N}_i=1}^{\infty} \psi_i^{\mathcal{N}_i} = \frac{\psi_i}{1-\psi_i}$$

yielding

$$E(\boldsymbol{\mathcal{N}_i}) = \frac{\psi_i}{1-\psi_i}$$

□

The parameter ψ_i can easily be calculated using lemma 1. Its value as well as the expected value of the time to failure and of the repair time depend on the distribution functions of $\boldsymbol{X}$ and $\boldsymbol{Y}$. We will now consider some possibilities.

Exponentially distributed failure and repair times: The results when $\boldsymbol{X}$ is exponentially distributed with parameter λ and $\boldsymbol{Y}$ is exponentially distributed with parameter μ are shown below.

First of all, we need to know the expected value of $E[\boldsymbol{\mathcal{N}_i}]$. Using equation 3.8 and lemma 1 we get:

$$E[\boldsymbol{\mathcal{N}_i}] = \frac{1-e^{-\lambda r_i d_i}}{e^{-\lambda r_i d_i}} \tag{3.10}$$

The expected value of the time to interruption is obtained using equation 3.6

and lemma 1:

$$\begin{aligned} E[\mathfrak{F}_i] &= \int_{\mathfrak{f}=0}^{d_i} \mathfrak{f} f'(\mathfrak{f}|\mathfrak{f} < d_i) d\mathfrak{f} \\ &= \frac{1}{\psi_i} \int_{\mathfrak{f}=0}^{d_i} \mathfrak{f} \lambda r_i e^{-\lambda r_i \mathfrak{f}} d\mathfrak{f} \\ &= \frac{1}{\lambda r_i} - \frac{d_i(1-\psi_i)}{\psi_i} \end{aligned} \tag{3.11}$$

The repair time is exponentially distributed with parameter μ so that:

$$E[\boldsymbol{Y}] = \frac{1}{\mu} \tag{3.12}$$

Observation 1. *Using the fact that the repair times are independently distributed we can write:*

$$\begin{aligned} E[\boldsymbol{\mathcal{Y}}_i] &= E[\boldsymbol{Y}] E[\boldsymbol{\mathcal{N}_i}] \\ &= \frac{\psi_i}{\mu(1-\psi_i)} \end{aligned}$$

This result can also be obtained by first determining the $\boldsymbol{\mathcal{N}_i}$-fold convolution of $\boldsymbol{Y}$ and then calculating the corresponding expected value.

Proof. In case the activity is interrupted $\boldsymbol{\mathcal{N}_i}$ times, the part of the duration extension attributable to repair times corresponds to an $\boldsymbol{\mathcal{N}_i}$-fold convolution of an exponential distribution with parameter μ. It can easily be shown that this convolution is gamma distributed with scale parameter μ and shape parameter $\boldsymbol{\mathcal{N}_i}$. This property allows us to write:

$$P(\boldsymbol{\mathcal{Y}_i} = y; \boldsymbol{\mathcal{N}_i}) = \frac{\mu^{\boldsymbol{\mathcal{N}_i}}}{\Gamma(\boldsymbol{\mathcal{N}_i})} y^{\boldsymbol{\mathcal{N}_i}-1} e^{-\mu y}$$

Let $P(\boldsymbol{\mathcal{Y}}_i = y)$ be the probability that the $\boldsymbol{\mathcal{N}_i}$-fold convolution of $\boldsymbol{Y}$ is

equal to y for all values of $\boldsymbol{\mathcal{N}_i}$. This probability can be calculated by averaging $P(y;\mathcal{N}_i)$ over the distribution of $\boldsymbol{\mathcal{N}_i}$. Because the Gamma distribution is not defined for shape parameters less than or equal to 0 and because this distribution would make no sense for $\mathcal{N}_i = 0$, we have to condition the probability distribution on the fact that at least one disruption occurs:

$$\begin{aligned} P(y) &= \frac{\sum_{\mathcal{N}_i=1}^{\infty} P(y;\mathcal{N}_i)h(\mathcal{N}_i)}{Pr\{\mathcal{N}_i > 0\}} \\ &= \frac{1}{\psi_i}\sum_{\mathcal{N}_i=1}^{\infty}\frac{\mu^{\mathcal{N}_i}}{\Gamma(\mathcal{N}_i)}y^{\mathcal{N}_i-1}e^{-\mu y}\psi_i^{\mathcal{N}_i}(1-\psi_i) \\ &= \frac{(1-\psi_i)e^{-\mu y}}{\psi_i y}\sum_{\mathcal{N}_i=1}^{\infty}\frac{\mathcal{N}_i(\mu y\psi_i)_i^{\mathcal{N}}}{\mathcal{N}_i!} \end{aligned}$$

using the property of the Gamma-function that $\Gamma(r+1) = r\Gamma(r)$ and $\Gamma(r+1) = r!$.

In order to find a closed-form expression for $P(y)$ we only need to find a closed-form expression for the infinite sum in the last equation. After substituting $\mu y\psi_i$ with α for ease of notation, it is easy to show that the infinite sum is equal to αe^{α} using MacLaurin series expansion:

$$f = \alpha e^{\alpha} = \sum_{\mathcal{N}_i=0}^{\infty}\frac{\alpha_i^{\mathcal{N}} f^{(\mathcal{N}_i)}(0)}{\mathcal{N}_i!}$$

because $f^{(\mathcal{N}_i)}(\alpha) = \mathcal{N}_i e^{\alpha} + \alpha e^{\alpha}$ and consequently $f^{(\mathcal{N}_i)}(0) = \mathcal{N}_i$.

This means that the probability density function of $\boldsymbol{\mathcal{Y}}_i$ can be written as:

$$\mathcal{G}(y) = (1-\psi_i)\mu e^{y\mu(\psi_i-1)}$$

with expected value:

$$\begin{aligned} E[\boldsymbol{\mathcal{Y}}_i] &= Pr(\boldsymbol{\mathcal{N}_i} = 0)0 + Pr(\boldsymbol{\mathcal{N}_i} > 0) \int_{y=0}^{\infty} (1-\psi_i) y\mu e^{y\mu(\psi_i - 1)} dy \\ &= \frac{\psi_i}{\mu(1-\psi_i)} \end{aligned}$$

Thanks to the convenient result that the n-fold convolution of an exponential distribution with parameter λ is gamma distributed with parameters n and λ and thanks to the easy derivation of the closed form expression of the infinite sum, we were able to determine a simple formula for the distribution function of $\boldsymbol{\mathcal{Y}}_i$. This will unfortunately not be possible for $\boldsymbol{\mathcal{X}}_i$ because of the difficulties in finding a usable expression for the n-fold convolution of a truncated distribution. Therefore, we only focus on expected value calculation in the rest of our analysis. □

Theorem 1. *In a preempt-repeat environment with fixed resource allocations, the expected duration extension due to breakdowns for an activity with duration d_i and resource usage r_i of a single renewable resource type for which the time to failure of each resource unit is exponentially distributed with parameter λ and the time to repair is exponentially distributed with parameter μ is given by:*

$$E[\boldsymbol{\sigma}_i] = \frac{\psi_i}{1-\psi_i}\left(\frac{1}{\lambda r_i} + \frac{1}{\mu}\right) - d_i \tag{3.13}$$

with $\psi_i = 1 - e^{-\lambda r_i d_i}$.

Proof. Substituting 3.10, 3.11 and 3.12 in 3.5 yields equation 3.13. □

Exponentially distributed failure times and uniformly distributed repair times: It is not always reasonable to assume that repair times are exponentially distributed. In some cases a uniform distribution might be

more suitable, the expected duration increase is then given by:

Theorem 2. *In a preempt-repeat environment with fixed resource allocations, the expected duration extension due to breakdowns for an activity with duration d_i and resource usage r_i of a single renewable resource type for which the time to failure of each resource unit is exponentially distributed with parameter λ and the time to repair is uniformly distributed between $[Y^{min}, Y^{max}]$ is given by:*

$$E[\boldsymbol{\sigma}_i] = \frac{\psi_i}{1-\psi_i}\left(\frac{1}{\lambda r_i} + \frac{Y^{min}+Y^{max}}{2}\right) - d_i \tag{3.14}$$

with $\psi_i = 1 - e^{-\lambda r_i d_i}$.

Proof. Substituting $E[\boldsymbol{Y}]$ with the expected value of the uniform distribution in 3.13 yields the above formula. □

3.1.2 Multiple resource types

The case of multiple resource types is somewhat more complicated. We still consider an activity i with a deterministic duration d_i. The only difference is that we now use R different resource types. In each time period of its execution, activity i requires r_{ik} units of resource type k. We extend the notation of the time to failure and the repair time with a subscript k to represent the considered resource type.

The main differences with the single resource type case are in the calculation of ψ_i and $E[\boldsymbol{\mathcal{Y}}_i]$.

The time to interruption of activity i ($\boldsymbol{\mathfrak{F}}_i$) is now determined by the minimum time to failure over all resource units over all resource types constrained between 0 and d_i. Let $\boldsymbol{X}_{mk}$ represent the time to failure of resource unit m of resource type k and let $\boldsymbol{X}_k^{min}$ represent $min(\boldsymbol{X}_{1k}, \boldsymbol{X}_{2k}, ..., \boldsymbol{X}_{r_{ik}k})$. Using lemma 1 we can write:

$$\begin{aligned}\psi_i &= Pr(\text{i interrupted}) \\ &= Pr[min(X_{11}, X_{21}, ..., X_{r_{i1}1}, ..., X_{1R}, X_{2R}, ..., X_{r_{iR}R}) \leqslant d_i] \\ &= 1 - [1 - F_1(d_i)]^{r_{i1}} ... [1 - F_R(d_i)]^{r_{iR}} \end{aligned} \tag{3.15}$$

Allowing for multiple resource types does not change the distribution function of $\boldsymbol{\mathcal{N}_i}$ so that equation 3.8 remains valid.

Deriving the expected value of $\mathfrak{R}_i$ is more complicated as the distribution function depends on the resource type that causes the disruption and that resource type will therefore determine the length of the downtime.

Exponentially distributed failure and repair times: If we let p_k represent the probability that the breakdown is caused by resource type k and that therefore the repair time will be exponentially distributed with parameter μ_k, we can write:

$$E[\mathfrak{R}_i] = \sum_k p_k \frac{1}{\mu_k} \tag{3.16}$$

The probability p_k is then the probability that the minimum time to failure over all resource units of resource type k is smaller than the minimum time to failure over all resource units of all resource types $l \neq k$. The following result is used in the calculation of p_k:

Lemma 3. *Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be independent stochastic variables that are both exponentially distributed, respectively with parameters λ and μ. The probability that $\boldsymbol{X}$ will be smaller than $\boldsymbol{Y}$ is then:*

$$Pr(\boldsymbol{X} < \boldsymbol{Y}) = \frac{\lambda}{\lambda + \mu} \tag{3.17}$$

We use lemma's 1 and 3 to determine p_k:

$$\begin{aligned} p_k &= Pr(\boldsymbol{X}_k^{min} < min_{l \neq k}(\boldsymbol{X}_l^{min})) \\ &= \frac{\lambda_k r_{ik}}{\lambda_k r_{ik} + \sum_{l \neq k} \lambda_l r_{il}} \\ &= \frac{\lambda_k r_{ik}}{\sum_l \lambda_l r_{il}} \end{aligned} \tag{3.18}$$

Theorem 3. *In a preempt-repeat environment with fixed resource allocations, the expected duration extension due to breakdowns for an activity with duration d_i and resource usage r_{ik} of renewable resource type k for which the time to failure of each resource unit is exponentially distributed with parameter λ_k and the time to repair is exponentially distributed with parameter μ_k is given by:*

$$\frac{\psi_i}{(1-\psi_i)(\sum_k \lambda_k r_{ik})} \left(1 + \sum_k \frac{\lambda_k r_{ik}}{\mu_k}\right) - d_i \tag{3.19}$$

with $\psi_i = 1 - e^{-d_i \sum_k \lambda_k r_{ik}}$.

Proof. Combining equations 3.8, 3.15, 3.16, 3.18 and equation 3.11 yields equation 3.19. □

Exponentially distributed failure times and uniformly distributed repair times: Again, we also consider the case of uniformly distributed repair times:

Theorem 4. *In a preempt-repeat environment with fixed resource allocations, the expected duration extension due to breakdowns for an activity with duration d_i and resource usage r_{ik} of renewable resource type k for which the time to failure of each resource unit is exponentially distributed with parameter λ_k and the time to repair is uniformly distributed between $[Y_k^{min}, Y_k^{max}]$ is given*

by:

$$\frac{\psi_i}{(1-\psi_i)(\sum_k \lambda_k r_{ik})}\left(1+\sum_k \frac{(Y_k^{min}+Y_k^{max})\lambda_k r_{ik}}{2}\right)-d_i \tag{3.20}$$

with $\psi_i = 1 - e^{-\lambda r_i d_i}$.

Proof. Substituting $E[\boldsymbol{Y}]$ with the expected value of the uniform distribution in 3.19 yields the above formula. □

3.2 Preempt-Resume

3.2.1 Single resource type

Again, due to breakdowns, the real duration of activity i is a stochastic variable $\boldsymbol{d}_i'$ consisting of a deterministic part d_i, corresponding to the duration of the activity when no interruption occurs and after which i is terminated, and a stochastic part $\boldsymbol{\sigma}_i$. The difference with the preempt-repeat case is that $\boldsymbol{\sigma}_i$ now only has to include the total repair time $\boldsymbol{\mathcal{Y}}_i$. If we preserve the notation of Section 3.1, equations 3.1 through 3.4 now become:

$$\boldsymbol{d}_i' = d_i + \boldsymbol{\sigma}_i \tag{3.21}$$

$$\boldsymbol{\sigma}_i = \boldsymbol{\mathcal{Y}}_i \tag{3.22}$$

$$\boldsymbol{\mathcal{Y}}_i = \boldsymbol{\mathfrak{R}}_{i1} + ... + \boldsymbol{\mathfrak{R}}_{i\boldsymbol{\mathcal{N}}_i} \tag{3.23}$$

From 3.5 we know that $E[\boldsymbol{\sigma}_i]$ can be calculated when we know the expected number of interruptions $E[\boldsymbol{\mathcal{N}_i}]$ and the expected repair duration $E[\boldsymbol{Y}]$. The expected number of interruptions can be calculated by dividing the duration of the activity i by the expected value of the time to interruption. The time to interruption is distributed according to the minimum of r_i independently and identically distributed variables $\boldsymbol{X}$. This distribution can be

determined using lemma 1.

Exponentially distributed failure and repair times:

Theorem 5. *In a preempt-resume environment with fixed resource allocations, the expected duration extension due to breakdowns for an activity with duration d_i and resource usage r_i of a single renewable resource type for which the time to failure of each resource unit is exponentially distributed with parameter λ and the time to repair is exponentially distributed with parameter μ is given by:*

$$E[\boldsymbol{\sigma}_i] = \frac{\lambda r_i d_i}{\mu} \tag{3.24}$$

Exponentially distributed failure times and uniformly distributed repair times:

Theorem 6. *In a preempt-resume environment with fixed resource allocations, the expected duration extension due to breakdowns for an activity with duration d_i and resource usage r_i of a single renewable resource type for which the time to failure of each resource unit is exponentially distributed with parameter λ and the time to repair is uniformly distributed between $[Y^{min}, Y^{max}]$ is given by:*

$$E[\boldsymbol{\sigma}_i] = \frac{\lambda r_i d_i (Y^{min} + Y^{max})}{2} \tag{3.25}$$

3.2.2 Multiple resource types

These results can easily be extended to the multiple resource type case in a similar fashion to the approach used in Section 3.1. When calculating the expected number of interruptions, we now need to consider the minimum time to failure over all resource units over all resource types. Furthermore, we should take care to weigh the mean time to repair with the probability that

the disruption is caused by a given resource type. This allows us to determine the expressions for the case of preempt-resume with multiple resource types.

Exponentially distributed failure and repair times: Using equation 3.16 and lemma 3 enables us to easily prove the following theorem:

Theorem 7. *In a preempt-resume environment with fixed resource allocations, the expected duration extension due to breakdowns for an activity with duration d_i and resource usage r_{ik} of renewable resource type k for which the time to failure of each resource unit is exponentially distributed with parameter λ_k and the time to repair is exponentially distributed with parameter μ_k is given by:*

$$E[\boldsymbol{\sigma}_i] \;=\; d_i \sum_k \frac{\lambda_k r_{ik}}{\mu_k} \tag{3.26}$$

Exponentially distributed failure times and uniformly distributed repair times:

Theorem 8. *In a preempt-resume environment with fixed resource allocations, the expected duration extension due to breakdowns for an activity with duration d_i and resource usage r_i of renewable resource type k for which the time to failure of each resource unit is exponentially distributed with parameter λ and the time to repair is uniformly distributed between $[Y_k^{min}, Y_k^{max}]$ is given by:*

$$E[\boldsymbol{\sigma}_i] \;=\; d_i \sum_k \frac{\lambda_k r_{ik}(Y_k^{min} + Y_k^{max})}{2} \tag{3.27}$$

3.3 Preempt-Setup

In the mixed form we assume that the reality lies somewhere in-between pure preempt-resume and pure preempt-repeat. Because it is hard to analytically evaluate the expected duration increase due to breakdowns, we chose to resort

to simulation. The problem lies in the fact that the time to failure is bounded by the remaining duration. In the preempt-repeat case this allows us to easily find an expression for ψ_i but unfortunately this is far more difficult here as the remaining duration is not constant but depends on the history of the breakdown process.

4 Time buffering

As we indicated in Section 2, our objective is to minimize the weighted sum of the expected absolute deviations between the actually realized and the planned activity starting times.

We assume that an initial precedence, resource and deadline feasible schedule S^u is given. One way to improve the objective function would then be to increase resource availability to such a high level that breakdowns having an impact on schedule feasibility become less likely (Lambrechts et al., 2007b). The cost-efficiency of this approach of course depends on the ratio of the instability costs versus the resource availability costs.

In the case that resources are fairly costly, it could be more interesting to simply accept the preemption of the affected activity but to try to mitigate the impact this has on the rest of the project network. This can be done by inserting explicit idle time into the schedule in such a way that the disruption due to the preemption of an interrupted activity does not translate into a starting time increase for non-preempted activities. Propagations of disruptions throughout the network depend first and foremost on the precedence structure of the network. However, the shared use of bottleneck resources should not be neglected. Consider the schedule depicted in Figure 2.

Activities 1, 2 and 3 are precedence unrelated activities with each a duration of 2 time units and a resource usage of respectively 4, 2 and 2 units of a single renewable resource type with a per period availability equal to 4. If we only consider precedence relations between activities, we would never insert

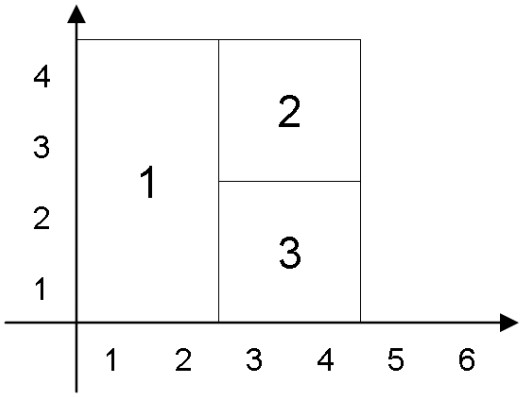

Figure 2: Why resource flow networks are indispensable for robustness estimation

any explicit idle time into this schedule because no activity has any impact on another activity according to this reasoning. However, this is clearly not the case in practice. If activity 1 is preempted, then clearly activities 2 and 3 would also need to be delayed since they need the 4 resource units that are used by activity 1 and that are only freed once activity 1 is completed.

Therefore, we use the approach introduced in section 2 in order to fix resource allocations during the time buffering phase. An additional advantage of this approach is that it greatly speeds up the calculation of a buffered schedule using a buffer list and the initial unbuffered schedule. The buffer list $\mathcal{B}$ indicates how many time units an activity should be started beyond its earliest precedence feasible starting time. The procedure for decoding a buffer list into a buffered schedule S^b is given in Algorithm 1.

Algorithm 1 Decoding procedure

1: $L = (i \in N$: ordered according to non-decreasing $s_i^u)$ (tie-break is lowest activity number)
2: $s_1^b = 0$
3: **for** $i := 2$ to n **do**
4: $\quad s_{L_i}^b = max(s_{L_i}^u, max_{j \in PRED_{L_i}}(s_{L_j}^b + d_{L_j})) + B_{L_i}$

In what follows, we present a steepest descent time buffering procedure that estimates the objective function value by means of simulation. However, because this approach is computationally quite demanding, we also present a heuristic that uses information regarding expected duration increases due to resource breakdowns that are calculated as shown in Section 3. Finally, since the procedures in Section 3 allow us to translate resource breakdowns into activity duration increases, it will become possible to use approaches developed for the resource-constrained project scheduling problem subject to stochastic activity durations. This means that we can use the effective and efficient STC-procedure developed by Van de Vonder et al. (2007b).

The project network shown in Figure 3 will be used to illustrate the algorithms described in this section. The example project consists of 10 activities with activity 1 and 10 respectively the dummy start and dummy end activities that are included to have a single starting and finishing node. The duration, the resource requirement of a single renewable resource type with a per period availability equal to 8 and the instability weight are shown above each activity. In this example we assume a project deadline of 18. The baseline starting time of the dummy start activity is then set to the release date of the project (time period 0), whereas the dummy end activity is assumed to end at the project deadline. Finally, we assume that each unit of the considered resource type has a mean time to failure equal to 30 and a mean time to repair equal to 3.

Artigues' resource allocation procedure yields the resource flow network $G = (N, A \cup A_R)$ that is shown in Figure 4.

4.1 Simulation-based steepest descent time buffering

A first possibility is to measure the quality of an intermediate, buffered schedule by means of simulation. In this simulation approach, the original project network is used (this means that we do not consider the extra resource arcs) and availability scenarios are generated using the mean times to failure and

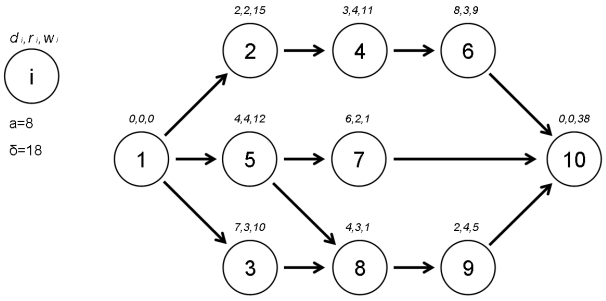

Figure 3: Example project network

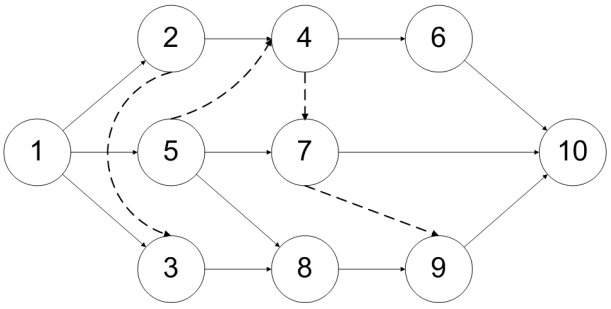

Figure 4: Resource flow network

the mean times to repair that are given in advance for each resource type. In order to speed up computation, a simple reactive policy is used. This *scheduled order* reactive policy reschedules activities in the order they were started in the baseline schedule while respecting precedence and resource constraints. It was shown by Lambrechts et al. (2007a) that this procedure performs relatively well considering the extremely low computational demands. Note that we chose to simulate project execution using the original project network because this enables us to obtain a better view about what is going to happen when we execute the project in practice. In practice, it would be very detrimental to the objective function value to simply use right-shift rescheduling (average instability values are on average a factor 20 higher than they would be when using for instance scheduled order rescheduling). Nevertheless, in the buffering step, the extended network is used in order to facilitate the insertion of time buffers into the schedule using the procedure in Algorithm 1.

The schedule is now iteratively buffered as follows. In each iteration every activity (except the dummy start activity) is considered for buffering. This activity is then right-shifted with one time unit. Affected activities are likewise right-shifted with one time unit in order to keep the schedule precedence feasible (and therefore also resource feasible because we keep resource flows fixed during time buffering) using the procedure in Algorithm 1. We buffer the activity leading to the highest improvement in the objective function value that yields a schedule respecting the deadline constraint. If no such activity can be found, the procedure is terminated. The pseudocode for this approach is given in Algorithm 2.

Algorithm 2 Time buffering heuristic

```
1: fix resource allocations using Artigues' procedure
2: z_best := evaluate(S)
3: while improvement found do
4:    activity := −1
5:    for i : 2 → n do
6:       B_i := B_i + 1
7:       z := evaluate(S)
8:       if s_n ⩽ δ AND z < z_best then
9:          z_best := z
10:         activity := i
11:      B_i := B_i − 1
12:   if activity ≠ −1 then
13:      B_activity := B_activity + 1
```

4.2 Time buffering using surrogate measures

Simulating the weighted instability objective function is computationally quite demanding. Even though this approach is feasible for small project networks, the computational demands quickly increase with project size to an unpractical level. This is why surrogate measures are necessary. These measures try to estimate the real instability costs and are calculated using information regarding activity characteristics and resource breakdown parameters.

In Section 3 it was shown how resource breakdowns can be translated into activity duration increases under various scenarios. This information will be used in this section to calculate two surrogate robustness measures. Furthermore, a third measure, based on the estimated probability that the start of an activity has to be postponed, is introduced.

The first surrogate objective ($SURR_1$) is calculated as follows:

$$SURR_1 = \sum_{j \in N} \sum_{i \in Pred_j^*} w_j max(0, s_i + d_i + LPL_{ij} + E[\boldsymbol{\sigma}_i] - s_j) \qquad (4.1)$$

For each activity j all predecessors, immediate as well as transitive ($i \in Pred_j^*$), are considered given the current extended network (strictly technical as well as resource arcs). For each such predecessor the expected impact of a duration increase of i on the starting time of j is calculated and these values are weighted with the instability weight of activity j and summed. In this equation, LPL_{ij} represents the length of the longest path between activities i and j. This longest path is determined based on the extended network and the given activity durations using a full enumeration approach.

The second surrogate objective ($SURR_2$) looks quite similar:

$$SURR_2 = \sum_{j \in N} max_{i \in Pred_j^*} w_j max(0, s_i + d_i + LPL_{ij} + E[\boldsymbol{\sigma}_i] - s_j) \qquad (4.2)$$

The main difference is that now the maximum starting time disruption for each activity j is calculated over all of its predecessors.

Finally, we consider a third measure ($SURR_3$) that is inspired on the starting time criticality heuristic (STC) developed by Van de Vonder et al. (2007b):

$$SURR_3 = \sum_{j \in N} STC_j \qquad (4.3)$$

The Starting Time Criticality (STC) heuristic is an elegant approach for generating time buffered schedules when faced with stochastic activity durations. It exploits information about the weights of the activities as well as about the probability distributions of the activity durations. The authors

define the starting time criticality of activity j as follows:

$$STC_j = w_j Pr(\mathbf{s}_j > s_j^0) \tag{4.4}$$

Using the observation that the starting time of activity j is disturbed whenever the duration increase of one of its predecessors i (be they immediate or transitive predecessors) is of such magnitude that it forces the delay of activity j in order to maintain precedence feasibility, the authors calculate the probability that activity j cannot start at its planned starting time s_j^0 as follows:

$$Pr(\mathbf{s}_j > s_j^0) = \sum_{i \in Pred_j^*} Pr(s_i + LPL_{ij} + \boldsymbol{d}_i > s_j) \tag{4.5}$$

assuming that predecessors start at their baseline starting times and that only one activity at a time disturbs $\mathbf{s}_j$. In case activity j is the dummy start activity or in case activity j's sole predecessor is the dummy start activity, the starting time criticality of j is equal to 0.

Because it is very hard to analytically calculate the probability distributions of the durations due to resource breakdowns, we approximate them by using simulation, assuming that resource allocations are fixed. For each activity 1000 simulation runs are executed to determine the probability for each possible duration outcome given the mean time to failure and the mean time to repair for each resource type used by that activity.

These surrogate measures are now used in the steepest descent approach of the previous paragraph (Algorithm 2) in order to evaluate the performance of the intermediate schedule. Note that we also tried to combine them with a tabu search improvement procedure but almost non-existing improvements did not seem to warrant the additional computational requirements.

We now illustrate the use of $SURR_1$ for generating a robust schedule starting from the minimal makespan schedule $S^u = (0, 0, 2, 4, 0, 7, 7, 9, 13, 18)$. Assuming a preempt-repeat scenario with exponentially distributed times to failure and repair times, we can calculate the expected duration increases

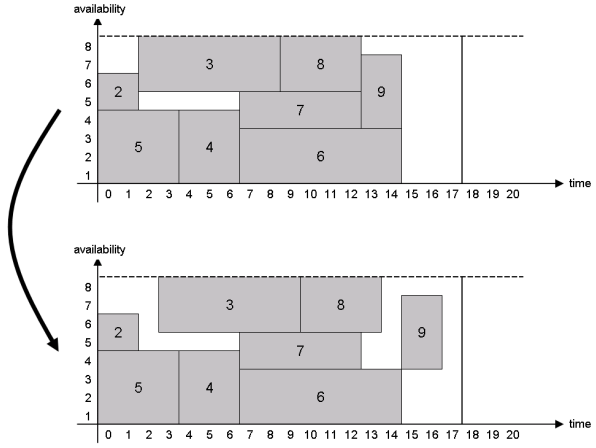

Figure 5: Initial and buffered schedule

using Theorem 1. This yields (rounded to the nearest integer): $E(\boldsymbol{\sigma}) = (0, 1, 6, 2, 3, 8, 3, 2, 1, 0)$.

In the unbuffered initial scheduling, buffering activities 2 through 9 yields respectively values for $SURR_1$ equal to $309, 299, 364, 385, 334, 345, 308, 304$. The lowest value is obtained when buffering activity 3 yielding the schedule $S = (0, 0, 3, 4, 0, 7, 7, 10, 14, 18)$. After this iteration step only activity 9 will be buffered with one time unit before terminating the procedure because no improving move can be found. The initial, unbuffered and the final, buffered schedule are both depicted in Figure 5.

4.3 Time buffering using the STC heuristic

In addition to the approaches presented in the previous two paragraphs we also implemented the STC heuristic developed by Van de Vonder et al. (2007b). The STC heuristic iteratively buffers the activity with the highest STC value so that the deadline constraint is respected until no more activities with an STC value larger than zero can be buffered without creating a deadline infeasible schedule. The STC values are calculated as shown in equations 4.4 and 4.5.

For more details regarding the operation of the STC heuristic we refer the reader to Van de Vonder et al. (2007b).

5 Computational Experiment

In order to test the relative performance of the various time buffering strategies we presented in this paper, we set up an extensive computational experiment. As a test set we used the 30-activity instances of the well known PSPLIB set of test instances (Kolisch and Sprecher, 1997). For each such instance we used 10 executions with different mean times to failure and mean times to repair. The mean times to failure were drawn from a uniform distribution between the minimal project makespan (obtained by solving the deterministic RCPSP) and two times this minimal makespan. The mean times to repair on the other hand were drawn from a uniform distribution between 1 and 5. As we indicated above, we assume that times to failure are drawn from an exponential distribution. Times to repair, on the other hand, can either be drawn from an exponential distribution with the mean time to repair as a parameter or from a uniform distribution defined between 50% and 150% of the mean time to repair. The instability weights were drawn from a triangularly shaped distribution between 1 and 10. The weight of the dummy end activity, however, was set to 10 times the average of this distribution in order to reflect the relatively higher importance of finishing the project in time than meeting individual milestones. Finally, the project deadline is set at the project's minimal makespan increased with 30%.

In our experiment, we first construct an initial schedule using either a procedure for optimally solving the deterministic RCPSP (Demeulemeester and Herroelen (1992) and Demeulemeester and Herroelen (1997)) or by generating a schedule in which activities that can be expected to have a high impact on the objective function value are scheduled as early as possible to prevent schedule perturbation. This strategy is called *highest CIW first* scheduling

and it is extensively described in Lambrechts et al. (2007b). Furthermore, it can be decided to include resource slack as described in Section 1. Resource buffering aims at scheduling the project using a reduced resource availability based on steady state probability calculations in order to prevent resource breakdowns from having an impact on activity starting times (Lambrechts et al., 2007b).

This initial schedule will then be buffered using one of the approaches described in Section 4. After generating the baseline schedule, the execution of this schedule will be simulated using real resource availabilities that are generated according to the chosen distribution functions for the times to failure and the times to repair. In case an infeasibility occurs, this infeasibility is resolved using the scheduled order reactive procedure described in Lambrechts et al. (2007a). Scheduled order was shown by Lambrechts et al. (2007a) to be an effective and very time-efficient way to resolve infeasibilities during schedule execution.

We consider three different breakdown settings. First of all, preempt-repeat as well as preempt-resume are considered. Furthermore, we also included a mixed form that allows continuation of execution from the point where the activity execution was interrupted at the expense of a certain setup time. We assume this setup time to be drawn from a uniform distribution defined over the interval $[0, 0.2d_i]$.

We applied each of our six time buffering procedures (no buffering (NT), simulated objective function value (sim), surrogate objective 1 ($SURR_1$), surrogate objective 2 ($SURR_2$), surrogate objective 3 ($SURR_3$) or the STC heuristic (STC)) to each of the four initial pre-schedule types (either minimal makespan (C_{max}) or highest CIW first (CIW) scheduling combined with either resource buffering (R) or no resource buffering (NR)). Each of those 24 strategies was tested for 10 different breakdown scenarios for each of the 480 test instances. Furthermore, 6 different environments were considered: preempt-repeat, preempt-resume or preempt-setup combined with either ex-

ponentially distributed or uniformly distributed repair times.

The aggregated results are shown in Figures 6 through 11. In each figure three values are given for each proactive policy type. First of all, we give the average instability value over all instances over all breakdown scenarios. Secondly, the average over all instances of the median instability values for the breakdown scenario are given. Finally, we also show the average of the worst case performances over all instances.

The added value of time buffering versus no time buffering immediately becomes apparent when we look at the six graphs. On average simulation-based time buffering performs more than four times better than no time buffering over all environments and over all initial schedule construction strategies. For the other time buffering methods this improvement lies between 1.5 and 2 times better, the best performer being STC and the worst performer being $SURR_2$. The same results, although even more pronounced, hold for median behavior. The improvement results for worst case performance are slightly less outspoken. When we look at individual environments, this observation is less conclusive for the case of preempt-setup in case a non-simulation-based approach is used together with resource buffering and a highest CIW first schedule. In those cases, the added value of time buffering is sometimes negligibly small.

Furthermore, as expected, time buffering based on a simulated objective function value always outperforms time buffering approaches that use surrogate objective function values. In this experiment we used 100 repetitions per schedule evaluation. An average improvement potential of at least a factor 2 is obtained when compared with each other time buffering procedure. This improvement almost doubles for the median values. Unfortunately, this comes at a quite heavy cost in terms of computation times. The average computation times are given in Figure 12. Simulation-based time buffering is on average a factor 10 slower than time buffering procedures using a surrogate objective function value. For our instances, these computation times are

still acceptable but this will not necessarily be the case for practical project networks consisting of 300 or more activities.

When looking at the average, median or worst-case behavior, averaged over all initial schedule strategies, we can draw some interesting conclusions. First of all, $SURR_2$ usually performs worst. The best performer is a bit harder to determine. In case of preempt-repeat, $SURR_1$ always performs best. In case of preempt-resume or preempt-setup, it seems preferable to use the STC heuristic. The performance difference is on average about 5%. The performance of $SURR_3$ usually lies somewhere in-between that of $SURR_1$ and STC.

As long as resource buffering nor time buffering are used, highest CIW first scheduling performs better than minimal makespan scheduling. This changes whenever a form of buffering is included. Furthermore, resource buffering always performs better than no resource buffering. This is especially noticeable if no time buffering is used. In that case, the performance difference is a factor 3 on average. This drops to a factor 1.25 for simulation-based time buffering and a factor of about 2 for other time buffering approaches. This is not surprising as good time buffering procedures should be mostly able to compensate for the deficiencies of suboptimal procedures for generating a robust initial schedule. Nevertheless, the results of resource buffering without time buffering are almost never outperformed by those of time buffering without resource buffering. One exception is the case of simulation-based time buffering which performs on average 2 times better than pure resource-based buffering.

Finally, we are having a closer look at the different environments we consider in our experiments. The results for the different strategies are comparable when either an exponential or a uniform repair time distribution is assumed. Assuming preempt-resume instead of preempt-repeat usually improves the average results by 50% or more. The results of preempt-setup when compared with preempt-repeat are, however, about a factor 2 worse on

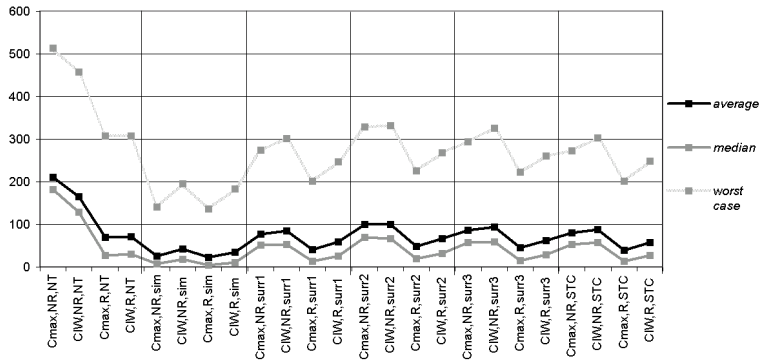

Figure 6: Aggregated results: exponential repair times and preempt-repeat

average. This is no doubt due to the fact that it is possible that an activity's duration can take on a very high value if it is frequently interrupted due to the incurring of setup times.

We did not yet extensively comment on the computation times. The computation times for the non-time buffered proactive strategies are negligibly small. As we stated before, they are quite high for simulation-based time buffering. Unsurprisingly, the results for $SURR_1$ and $SURR_2$ are comparable to those for $SURR_3$ and STC. The computation times of the latter two strategies are on average 16% higher than those of the former two. This is no doubt due to the computationally more demanding calculation of the STC values in each evaluation step.

6 Conclusions

We can conclude that time buffering is a very interesting alternative for incorporating robustness into a schedule. We gave an overview of analytical approaches for determining the expected duration increase an activity experiences due to resource breakdowns. Those results are used to create an effective and efficient algorithm for inserting explicit idle time into an initial, unbuffered schedule in order to protect it from the propagation of disrup-

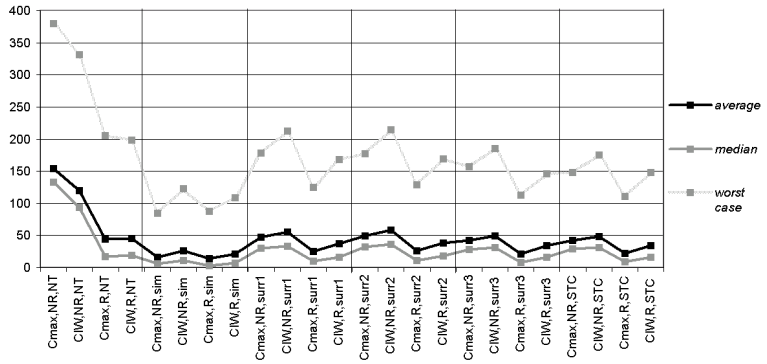

Figure 7: Aggregated results: exponential repair times and preempt-resume

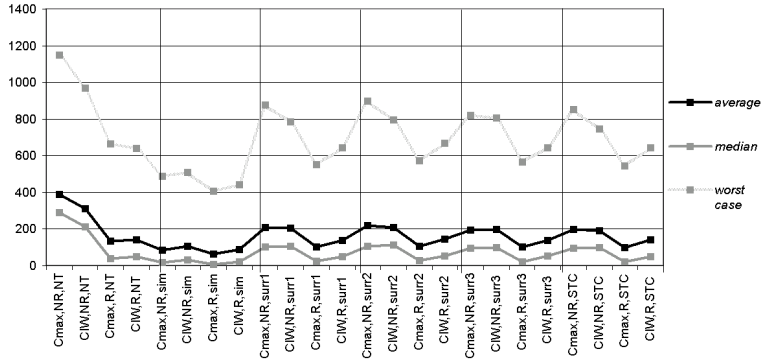

Figure 8: Aggregated results: exponential repair times and preempt-setup

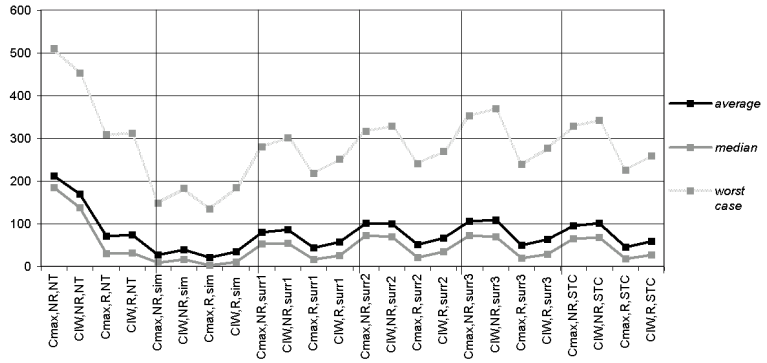

Figure 9: Aggregated results: uniform repair times and preempt-repeat

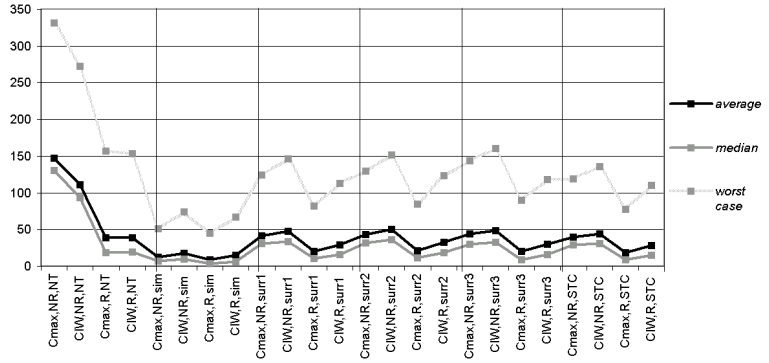

Figure 10: Aggregated results: uniform repair times and preempt-resume

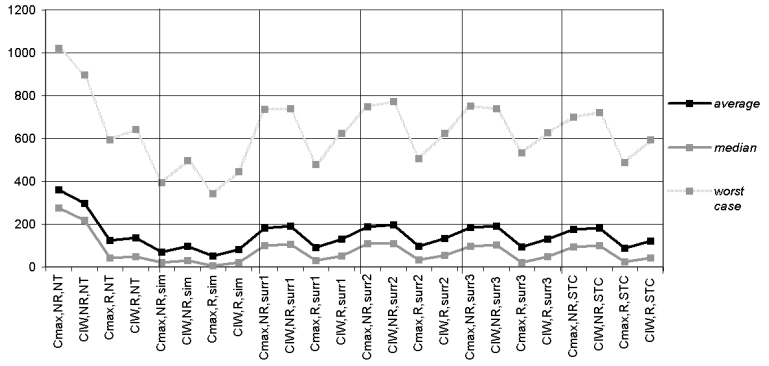

Figure 11: Aggregated results: uniform repair times and preempt-setup

computation time (sec)			exponential & repeat	exponential & resume	exponential & setup	uniform & repeat	uniform & resume	uniform & setup	average
NT	NR	Cmax	0,07	0,07	0,03	0,03	0,03	0,03	*0,04*
		CIW	0,00	0,00	0,00	0,00	0,00	0,00	*0,00*
	R	Cmax	0,10	0,10	0,04	0,04	0,04	0,04	*0,06*
		CIW	0,00	0,00	0,00	0,00	0,00	0,00	*0,00*
sim	NR	Cmax	4,85	4,39	4,62	3,93	3,43	4,58	*4,30*
		CIW	3,77	3,37	3,81	3,05	2,64	3,60	*3,37*
	R	Cmax	2,63	2,11	2,73	2,16	1,64	2,64	*2,32*
		CIW	2,33	1,92	2,51	1,98	1,58	2,41	*2,12*
surr1	NR	Cmax	0,15	0,15	0,21	0,17	0,16	0,21	*0,18*
		CIW	0,33	0,33	0,43	0,36	0,36	0,41	*0,37*
	R	Cmax	0,24	0,24	0,32	0,27	0,27	0,32	*0,28*
		CIW	0,42	0,42	0,53	0,51	0,51	0,56	*0,49*
surr2	NR	Cmax	0,15	0,15	0,21	0,17	0,16	0,21	*0,18*
		CIW	0,33	0,33	0,43	0,36	0,36	0,41	*0,37*
	R	Cmax	0,24	0,24	0,32	0,27	0,27	0,32	*0,28*
		CIW	0,43	0,42	0,53	0,52	0,51	0,56	*0,49*
surr3	NR	Cmax	0,28	0,22	0,22	0,24	0,21	0,21	*0,23*
		CIW	0,46	0,40	0,43	0,44	0,40	0,41	*0,42*
	R	Cmax	0,37	0,32	0,32	0,35	0,31	0,32	*0,33*
		CIW	0,56	0,50	0,54	0,59	0,56	0,56	*0,55*
STC	NR	Cmax	0,28	0,22	0,21	0,24	0,21	0,21	*0,23*
		CIW	0,46	0,40	0,43	0,44	0,40	0,41	*0,42*
	R	Cmax	0,37	0,31	0,32	0,34	0,31	0,32	*0,33*
		CIW	0,55	0,50	0,53	0,59	0,56	0,56	*0,55*
	average		*0,81*	*0,71*	*0,82*	*0,71*	*0,62*	*0,80*	

Figure 12: Average computation times

tions throughout the project network. It was shown that time buffering based on simulation performs far better than surrogate objective functions, but the reader should keep the higher computational demands in mind. Especially in practical project scheduling those computational demands will often become prohibitive. Therefore we suggest to either implement time buffering based on the first surrogate objective function or using the *STC* heuristic. *STC* offers the additional advantage that it has proven to be a good buffering strategy in case stochastic activity durations are considered. It would therefore be an interesting topic for further research to develop an integrated approach combining uncertain activity durations with unexpected machine breakdowns. The advantages of robust project scheduling for practical project management are obvious. Less rescheduling and replanning allows for a decrease in the costs resulting from those actions. Furthermore, the project manager will be able to quote reliable milestone delivery dates facilitating negotiations with customers and sub-contractors.

References

Artigues, C. and Roubellat, F. (2000). A polynomial activity insertion algorithm in a multi-resource schedule with cumulative constraints and multiple modes. *European Journal of Operational Research*, 127, pp 294–316.

Barlow, R. and Proschan, F. (1996). *Mathematical Theory of Reliability*. John Wiley & Sons Inc, New York.

Blumenfeld, D. (2001). *Operations research calculations handbook*. CRC Press LLC, Florida.

Brucker, P., Drexl, A., Möhring, R., Neumann, K. and Pesch, E. (1999). Resource-constrained project scheduling: Notation, classification, models and methods. *European Journal of Operational Research*, 112, pp 3–41.

Davenport, A. and Beck, J. (2002). A survey of techniques for scheduling with uncertainty. *Available from http://tidel.mie.utoronto.ca/publications.php*.

Demeulemeester, E. and Herroelen, W. (1992). A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science*, 38, pp 1803–1818.

Demeulemeester, E. and Herroelen, W. (1997). New benchmark results for the resource-constrained project scheduling problem. *Management Science*, 43, pp 1485–1492.

Demeulemeester, E. and Herroelen, W. (2002). *Project scheduling - A research handbook*. Vol. 49 of *International Series in Operations Research & Management Science*. Kluwer Academic Publishers, Boston.

Girault, M. (1959). *Initiation aux Processus Aléatoires*. Dunod, Paris.

Herroelen, W., De Reyck, B. and Demeulemeester, E. (1998). Resource-constrained scheduling: A survey of recent developments. *Computers and Operations Research*, 25, pp 279–302.

Herroelen, W. and Leus, R. (2004). The construction of stable project baseline schedules. *European Journal of Operational Research*, 156(3), pp 550–565.

Hopp, W. and Spearman, M. (2001). *Factory physics: Foundations of manufacturing management*. McGraw-Hill.

Kolisch, R. and Sprecher, A. (1997). PSPLIB - A project scheduling library. *European Journal of Operational Research*, 96, pp 205–216.

Lambrechts, O., Demeulemeester, E. and Herroelen, W. (2007a). Exact and suboptimal reactive strategies for resource-constrained project scheduling with uncertain resource availabilities. *FETEW Research Report KBI0702, K.U.Leuven, 36 pp.*

Lambrechts, O., Demeulemeester, E. and Herroelen, W. (2007b). Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities. *Journal of Scheduling, to appear*.

Leus, R. (2004). The generation of stable project plans. *4OR, The Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 2(3), pp 251–254.

Leus, R. and Herroelen, W. (2004). Stability and resource allocation in project planning. *IIE Transactions*, 36(7), pp 667–682.

O'Donovan, R., Uzsoy, R. and McKay, K. (1999). Predictable scheduling of a single machine with breakdowns and sensitive jobs. *International Journal of Production Research*, 37, pp 4217–4233.

Van de Vonder, S. (2006). *Proactive-reactive procedures for robust project scheduling*. PhD thesis. Faculuty of applied economics, K.U. Leuven, Belgium.

Van de Vonder, S., Ballestin, F., Demeulemeester, E. and Herroelen, W. (2007a). Heuristic procedures for reactive project scheduling. *Computers & Industrial Engineering*, 52(1), pp 11–28.

Van de Vonder, S., Demeulemeester, E. and Herroelen, W. (2007b). Proactive heuristic procedures for robust project scheduling: an experimental analysis. *European Journal of Operational Research (to appear)*, .

Van de Vonder, S., Demeulemeester, E. and Herroelen, W. (2007c). An investigation of efficient and effective predictive-reactive project scheduling procedures. *Journal of Scheduling (Special Issue on Project Scheduling under Uncertainty (Demeulemeester, E.L. and Herroelen W.S. (eds.))*, 10(2), pp 11–28.

Van de Vonder, S., Demeulemeester, E., Herroelen, W. and Leus, R. (2005). The use of buffers in project management: The trade-off between stability and makespan. *International Journal of Production Economics*, 97(2), pp 227–240.

Van de Vonder, S., Demeulemeester, E., Herroelen, W. and Leus, R. (2006). The trade-off between stability and makespan in resource-constrained project scheduling. *International Journal of Production Research*, 44(2), pp 215–236.

Yu, G. and Qi, X. (2004). *Disruption Management - Framework, models and applications*. World Scientific, New Jersey.